Herramienta para facilitar el desarrollo de aplicaciones basadas en Kinect

Rodrigo Ibañez, Damián Fanaro

Director: Dr. Álvaro Soria

Instituto de Investigación ISISTAN, Facultad de Cs. Exactas, UNCPBA
Campus Universitario, Paraje Arroyo Seco, CP B7001BBO, Tandil, Bs. As., Argentina
rodrigo.ibanez@isistan.unicen.edu.ar, damianfanaro@gmail.com
alvaro.soria@isistan.unicen.edu.ar

Resumen. El abaratamiento de los costos en los dispositivos de interacción natural ha promovido la investigación y el desarrollo de nuevas aplicaciones basadas en gestos. El dispositivo más popular es Microsoft Kinect el cual permite el seguimiento de los movimientos de los usuarios. Si bien Kinect facilita la inclusión de interacción natural en aplicaciones, el reconocimiento de gestos y la ejecución de las correspondientes acciones en la aplicación siguen siendo problemas aún no resueltos En este trabajo se presenta una herramienta que facilita el desarrollo de aplicaciones que interactúan por medio de gestos basadas en Kinect. La herramienta provee técnicas de aprendizaje de máquina que reconocen gestos a partir de las posiciones en el espacio de las partes del cuerpo provistas por Kinect. Las evaluaciones de la herramienta mostraron que la precisión del reconocimiento alcanza el 97% de los gestos evaluados y logra una reducción de un 67% del esfuerzo de desarrollo en términos de líneas de código.

1. Introducción

La utilización de dispositivos de Interfaz Natural (NUI) para el control de sistemas mediante gestos se ha vuelto algo común en nuestra vida cotidiana [1,2]. La aparición de Wiimote a finales de 2006 cambió el concepto de control remoto, permitiendo detectar los movimientos de la mano en un espacio 3D [3]. Aumentando la apuesta, Microsoft lanzó el dispositivo Kinect en el año 2010, el cual no sólo permite el seguimiento de la mano sino que también es capaz de identificar un conjunto de partes del cuerpo en tiempo real. Este dispositivo de bajo costo ha facilitado el desarrollo de aplicaciones orientadas a NUI dando lugar a un nuevo abanico de aplicaciones [4]. El Software Development Kit[1] (SDK) desarrollado por Microsoft para Kinect brinda interfaces para el acceso a diferentes datos provistos por el dispositivo, tales como imágenes RGB, imágenes de profundidad y la posición en el espacio de 20 partes del cuerpo sin necesidad de implementar software adicional.

[1] http://www.microsoft.com/en-us/kinectforwindows/

Sin embargo, el SDK aún carece de soporte para el reconocimiento de gestos, entendiendo por gesto al movimiento realizado por las partes del cuerpo con la intensión de ejecutar una acción. En consecuencia, si un desarrollador desea implementar una aplicación de interfaz natural con Kinect tendrá que enfocarse también en implementar un mecanismo de reconocimiento de gestos. Para alcanzar este objetivo, algunos enfoques proponen la definición de reglas sobre las posiciones de las partes del cuerpo permitiendo reconocer posturas estáticas o movimientos simples del cuerpo [5,6]. El problema de este tipo de enfoque es que se ve limitado al momento de considerar ciertos aspectos de los gestos tales como: la variación de la altura y posición de los usuarios dentro del campo de detección, las diferencias de velocidad de los usuarios al momento de realizar el gesto, los diferentes niveles de destreza con la que los usuarios realizan los gestos y la dificultad para definir reglas para el reconocimiento de gestos complejos [5, 6, 7]. A estos aspectos se les suma la problemática de la administración de los gestos y sus reconocedores, ya que sería interesante poder reutilizar los reconocedores de gestos en diferentes aplicaciones reduciendo de esta manera el esfuerzo de su especificación.

Abordar esta problemática con enfoques que utilizan técnicas como *Supervised Machine Learning* (en español aprendizaje de máquinas supervisado) [8] aparece como una alternativa adecuada. Estas técnicas utilizan una base de gestos de entrenamiento para construir un clasificador. Este clasificador evalúa los movimientos del usuario y determina su similitud con cada gesto entrenado. Si bien estas soluciones ofrecen reconocimientos cercanos al 100% de los gestos evaluados, el desarrollador se ve obligado a implementar técnicas complejas para poder incluir el reconocimiento de gestos dentro de su aplicación.

En este contexto, se presenta *GRTool* (por *Gesture Recognition Tool*, en español Herramienta de Reconocimiento de Gestos); una herramienta que facilita la definición y reconocimiento de gestos con Kinect. *GRTool* permite al desarrollador construir un conjunto de gestos entrenamiento y provee soporte para construir los clasificadores que identifiquen gestos similares a aquellos entrenados. Para la especificación del conjunto de gestos de entrenamiento, *GRTool* brinda una interfaz gráfica que permite la grabación, edición y administración de gestos utilizando el dispositivo Kinect.

Teniendo el conjunto de gestos modelo (llamado base de gestos de entrenamiento en *GRTool*), *GRTool* permite el entrenamiento de diferentes técnicas de *Machine Learning* para el reconocimiento de gestos. En particular, *GRTool* soporta tres técnicas: Dynamic Time Warping [9], Hidden Márkov Models [10] y Procrustes Analysis [11].

Con el objetivo de evaluar *GRTool*, se analizaron dos aspectos: la precisión de las técnicas de reconocimiento y los esfuerzos que requiere desarrollar una aplicación con y sin *GRTool*. Para evaluar la precisión del reconocimiento de gestos de *GRTool*, se probaron las tres técnicas con un conjunto de 4 gestos distintos con 100 ejemplares para cada gesto. Los resultados mostraron reconocimientos superiores al 95% de los gestos evaluados.

Con respecto al esfuerzo requerido para la implementación de aplicaciones de interfaz natural con *GRTool*, se utilizó una aplicación que permite manipular objetos 3D por medio de 5 gestos implementada directamente sobre la interfaz del SDK y se

comparó su implementación con una nueva versión implementada por medio de *GRTool*. Los resultados de esta comparación mostraron que la versión con *GRTool* requirió un 69,63% menos de líneas de código totales y una disminución del 67,04% en la complejidad ciclomática del código con respecto a la implementación de la aplicación sobre el SDK de Kinect.

El resto del artículo se organiza de la siguiente manera: la Sección 2 describe el funcionamiento y utilización de *GRTool*. La Sección 3 describe dos experimentos y los resultados de evaluar no sólo la precisión de *GRTool* sino los beneficios de su aplicabilidad. Finalmente, las conclusiones son presentadas en la Sección 4.

2 Enfoque para el reconocimiento de gestos

Cuando una persona se sitúa frente a Kinect, el SDK identifica la posición de 20 partes de su cuerpo en el espacio 3D (X, Y, Z). Estos puntos en el espacio son actualizados por el SDK 30 veces por segundo y agrupados en frames permitiendo a las aplicaciones manipular dicha información. Si estas posiciones son observadas durante un intervalo de tiempo se obtienen los movimientos de las partes del cuerpo, es decir, el gesto que realizó la persona.

El objetivo principal de este artículo es describir *GRTool*; una herramienta que facilita el reconocimiento de gestos soportado por técnicas de *Machine Learning*. Las técnicas provistas por *GRTool* utilizan un aprendizaje supervisado sobre un conjunto de gestos de entrenamiento clasificados, que forma la base del conocimiento, para clasificar o reconocer nuevos gestos. La Fig. 1 muestra un esquema conceptual del enfoque propuesto donde se describe la interacción de *GRTool* con una aplicación.

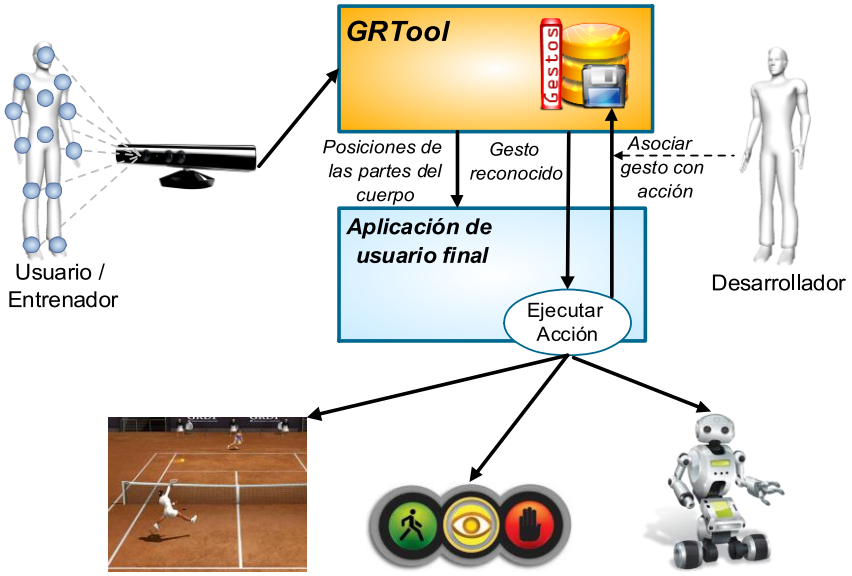


Fig. 1. Esquema conceptual de *GRTool*

GRTool es la encargada de reconocer un gesto a partir de la información de movimientos aportada por Kinect y notificar su reconocimiento a la aplicación por medio de eventos. Estos eventos están asociados a la realización de una acción específica dentro de la aplicación desarrollada, pudiendo ser la ejecución de un movimiento dentro de un juego, una alerta de seguridad sobre movimientos sospechosos o manipulación de robots, entre otros. Además, *GRTool* provee en todo momento las posiciones de las partes del cuerpo del usuario permitiendo a la aplicación actualizar la representación virtual del mismo mediante un avatar.

Para lograr este objetivo *GRTool* define tres pasos. El primer paso consiste en la construcción de una base de gestos de entrenamiento que contenga los gestos modelos a reconocer. Sobre esta base de entrenamiento, *GRTool* es capaz de entrenar las técnicas de *Machine Learning* construyendo los clasificadores de gestos. Por último, *GRTool* provee un conjunto de interfaces para asociar cada uno de los gestos a ser reconocido por los clasificadores con una acción de ejecución dentro de la aplicación de usuario final. Las siguientes secciones se detallan cada uno de estos pasos.

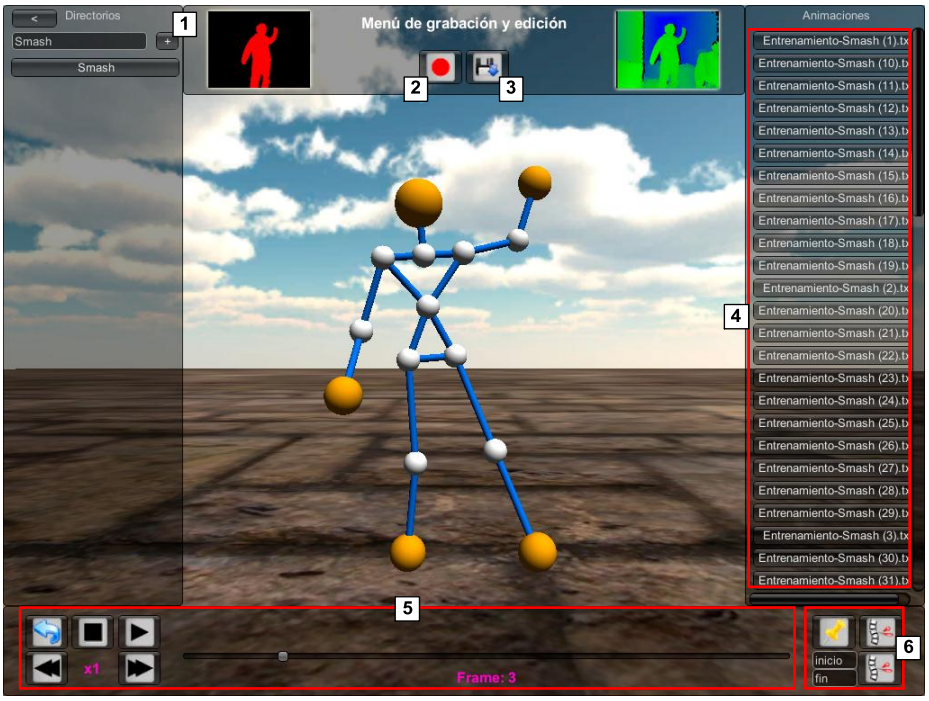
2.1 Preparación de la base de gestos de entrenamiento.

Con el fin de facilitar la comprensión del funcionamiento de *GRTool* se utilizará un ejemplo conductor. El ejemplo consiste en el desarrollo de una aplicación que simula un partido de tenis y que debe reconocer el gesto *Smash* (Fig. 2). Un *Smash* consiste en un golpe de tenis que se da a la pelota desde arriba hacia abajo por encima de la altura de la cabeza, para lanzarla violentamente contra el campo del rival.

Fig. 2. Ejecución del gesto *Smash* en una secuencia de seis frames

El primer paso que debe realizar el desarrollador para incorporar el reconocimiento de gestos de *GRTool* es generar un conjunto de entrenamiento con los movimientos del *Smash*. Para esto, *GRTool* posee una interfaz para la administración de la base de gestos de entrenamiento (Fig. 3). El proceso de generación del entrenamiento inicia con la definición del nombre del gesto que se va a almacenar (1) en el panel lateral izquierdo. A continuación, el entrenador presiona el botón de grabar (2) en el panel superior central y luego de 5 segundos, tiempo necesario para permitir al entrenador posicionarse frente a Kinect, *GRTool* comienza a capturar los movimientos. En este punto, el entrenador realiza el gesto y al finalizar presiona el botón de guardar (3)

haciendo que el *Smash* sea almacenado. Esto produce una nueva entrada en la lista del panel lateral derecho (4). Cada gesto almacenado contiene los frames obtenidos por Kinect correspondientes a los movimientos realizados por el entrenador. Es importante notar que este proceso puede ser repetido varias veces para generar variantes del mismo gesto tanto en la forma de ejecutarlo en diferentes puntos del campo de detección como en diferentes contexturas físicas variando la persona que realiza el gesto. Contar con variantes del gesto, permite a las técnicas de reconocimiento considerar diferentes ejemplos del gesto y consecuentemente mejorar su precisión.



Referencias:

1- Administración de gestos	4- Listado de ejecuciones del gesto
2- Inicio y detención de la grabación	5- Control de reproducción de gesto
3- Almacenar gesto	6- Control de edición de gesto

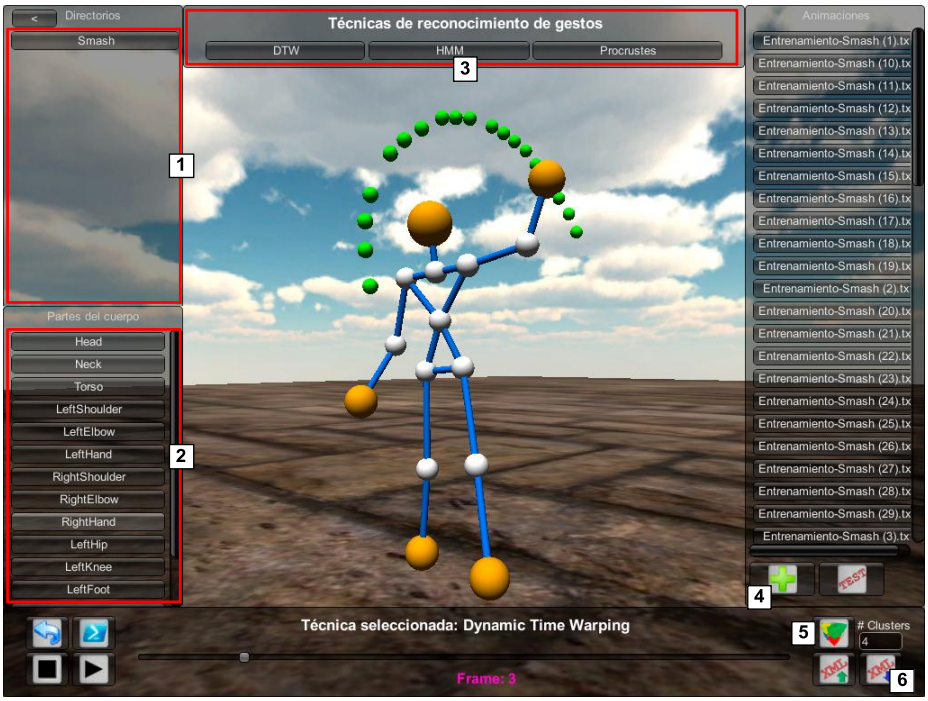
Fig. 3. Interfaz gráfica de *GRTool* para preparar la base de gestos de entrenamiento

Al finalizar el proceso de almacenamiento de los movimientos para cada gesto, el desarrollador debe validar la calidad de los gestos capturados. En este punto se debe depurar el conjunto de entrenamiento ya que no sólo contiene los movimientos relevantes al gesto sino que también incluye aquellos movimientos de posturas iniciales o finales que no corresponden al gesto. Estos movimientos pueden influir negativamente en la precisión de las técnicas de reconocimiento. Para esto, *GRTool* provee un panel de control de reproducción y edición de gestos que se muestra en la Fig. 3. El

proceso de validación de gestos comienza seleccionando el gesto a validar del listado de ejecuciones del gesto (4). En este punto, la ejecución del gesto seleccionado se visualiza mediante el avatar en el panel central y la reproducción del mismo se controla utilizando los botones de reproducción (5). Mediante esta interfaz, el desarrollador identifica los frames que no corresponden al gesto *Smash* y los elimina utilizando el panel de edición (6). Para esto, el desarrollador cuenta con dos opciones: ingresar el número de frame inicial y final en los campos de texto o insertar marcas a medida que el gesto se reproduce. Al finalizar presiona el botón recortar y el gesto modificado es almacenado. Luego de editar todas las ejecuciones del gesto *Smash*, se obtiene la base de gestos de entrenamiento validada.

2.2 Entrenamiento de técnicas de reconocimiento de gestos

Contando con la base de gestos de entrenamiento del *Smash*, el desarrollador debe entrenar alguna de las tres técnicas de reconocimiento provistas por *GRTool*. La interfaz provista por *GRTool* para el entrenamiento de las técnicas se muestra en Fig. 4. Inicialmente, el entrenador debe seleccionar el gesto a entrenar en el panel lateral izquierdo (1) y a continuación las trayectorias de las partes de cuerpo que considere relevantes en el reconocimiento del mismo (2). Por simplificación, supóngase que en nuestro ejemplo la trayectoria de la mano derecha es suficiente para determinar el gesto *Smash*. Tomando como base la mano derecha, se selecciona la técnica deseada en el panel superior (3) y se inicia el entrenamiento con el botón entrenar (4).



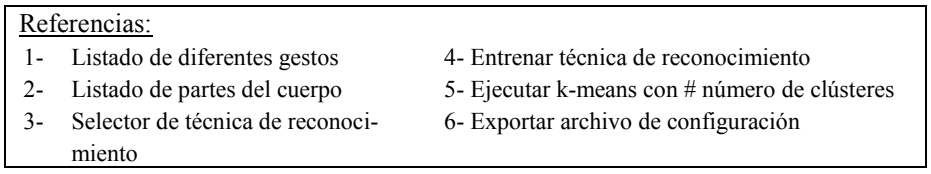


Fig. 4. Interfaz gráfica de *GRTool* para entrenar las técnicas de reconocimiento de gesto

El primer paso del proceso de entrenamiento realiza una simplificación para facilitar la comparación entre trayectorias, por medio de un pre-procesado sobre las trayectorias de la mano derecha de todo el conjunto de entrenamiento del *Smash*. Este pre-procesado consiste en trasladar individualmente cada trayectoria en el conjunto de entrenamiento al centro de coordenadas. Una vez finalizado el pre-procesado, cada técnica manipula convenientemente el conjunto de entrenamiento para generar un valor que indique el umbral de aceptación para el gesto. Una breve descripción de cada técnica y el umbral generado se presenta a continuación:

- Dynamic Time Warping (DTW): El algoritmo DTW [9] busca una alineación óptima entre dos secuencias temporales y en principio fue ideado para encontrar patrones semejantes entre secuencias de sonido. En este trabajo, esas secuencias están formadas por puntos en el espacio que corresponden a la trayectoria descrita por la mano derecha. El algoritmo alinea dos trayectorias estirando y encogiendo el eje temporal iterativamente hasta lograr una mínima distancia entre cada par de puntos de las trayectorias. El resultado de aplicar DTW es un valor de distancia que permite calcular cuánto se asemejan dos trayectorias. Luego de aplicarlo sobre todo el conjunto de entrenamiento del *Smash* se obtiene un valor de referencia que marca el límite superior del umbral de aceptación para el gesto. Cuando un nuevo gesto intente ser reconocido, si el valor de aplicar DTW sobre la trayectoria de la mano derecha es menor al valor de referencia, el gesto será aceptado como válido.
- Hidden Márkov Models (HMM): Estos modelos pueden ser vistos como una máquina finita de estados, compuesta de estados ocultos y de estados observables asociados a dichos estados ocultos. El objetivo de esta técnica es descubrir el estado oculto en el que se encuentra el modelo a parir de los valores de los estados observables. Para aplicar HMM [10], en este trabajo se convierten los puntos de la trayectoria de la mano derecha a una cantidad finita de estados utilizando el algoritmo de k-means [12]. Este algoritmo agrupa los puntos de la trayectoria, usando el criterio de cercanía, en un conjunto de clústeres numerados convirtiendo el *Smash* en una secuencia numérica equivalente que es utilizada como entrada para entrenar HMM. En HMM, la cantidad de clústeres, es igual a la cantidad de estados ocultos y a la cantidad de estados observables del modelo. El resultado de aplicar HMM es un valor que mide la probabilidad de que el gesto pertenezca al modelo entrenado. Luego de aplicarlo sobre todo el conjunto de entrenamiento del *Smash* se obtiene un valor de referencia que marca el límite inferior del umbral de aceptación para el gesto. Esto se debe a que aquí se busca una mayor probabilidad mientras que en DTW se busca una menor distancia. Entonces, cuando un nuevo gesto intente ser

reconocido, si el valor de aplicar HMM sobre la trayectoria de la mano derecha es mayor al valor de referencia el gesto será aceptado como válido.

- Procrustes Analysis: El análisis de Procrustes [11] busca una superposición óptima entre dos figuras. Para esto, aplica una serie de transformaciones matemáticas que modifican las figuras y asegura obtener una mínima distancia entre ellas. Para aplicar el análisis de Procrustes en el contexto del reconocimiento de gestos se ve a las trayectorias de la mano derecha como figuras estáticas. El primer paso del algoritmo es normalizar las trayectorias para llevar todos sus puntos a valores entre cero y uno. En este punto, las trayectorias se encuentran superpuestas debido al preprocesado de trasladar y normalizar. Luego, el algoritmo rota las trayectorias hasta obtener una mínima distancia entre ellas. El resultado de aplicar el análisis de Procrustes es un valor de distancia que mide cuánto se asemejan dos trayectorias y se utiliza de la misma manera que en DTW.

En este punto, el desarrollador ha entrenado las técnicas de reconocimiento y obtenido un valor de referencia que marca el umbral de aceptación para el gesto. Si la técnica entrenada es DTW o Procrustes este valor de referencia indica una distancia mientras que si es HMM indica una probabilidad. En el primer caso, los gestos de entrada deben generar una distancia menor al valor de referencia para ser reconocidos mientras que en el segundo deben generar una probabilidad mayor. Cada técnica presentada provee características particulares, ventajas y desventajas. La selección de cuál usar dependerá de las necesidades del desarrollador. Por ejemplo, si se busca una alta exactitud de reconocimiento, en términos de la calidad del gesto imitado, entonces se debe usar HMM. Mientras que si busca obtener más información sobre cómo el gesto se está reconociendo, el desarrollador deberá inclinarse por DTW o Procrustes y así obtener parámetros como la escala o rotación aplicada al gesto para que sea reconocido.

Una vez finalizado el entrenamiento de la técnica seleccionada, el último paso consiste en almacenar el resultado de dicho entrenamiento para su posterior uso dentro de la aplicación de usuario final. Para esto, el desarrollador presiona el botón *exportar XML* (6) en Fig. 4 y *GRTool* genera un archivo de configuración que contiene el tipo de técnica utilizada, las partes del cuerpo que se tienen en cuenta para el reconocimiento del gesto y los valores de umbral generados entre otros valores (Ver más detalles sobre el XML en el Apéndice A).

2.3 Especificación y reconocimiento de gestos

Finalizado el entrenamiento del *Smash*, el desarrollador debe asociar el reconocimiento del gesto a una acción en la aplicación. Para esto, supóngase que nuestra aplicación que simula un partido de tenis es desarrollada en lenguaje C# y utilizando el motor gráfico Unity3D[2]. En este contexto, el desarrollador debe incorporar el Algoritmo 1 dentro del código de su aplicación. Este algoritmo respeta la estructura de un

[2] http://spanish.unity3d.com/

script por defecto dentro de Unity3D, definiendo dos estados en la ejecución de la aplicación: inicialización (método *Start*) y actualización (método *Update*).

La inicialización representada por el método *Start* (línea 1) es ejecutada una única vez por Unity3D y tiene como objetivo definir las variables que serán utilizadas durante el resto de la ejecución de la aplicación. En este método, el primer paso es definir el usuario que *GRTool* observará y sobre el cual se realizará el reconocimiento de gestos (línea 2). Con esta instrucción, el desarrollador obtiene la referencia al esqueleto del usuario 1 provisto por *GRTool*. Este esqueleto, es una estructura que contiene la posición en el espacio de cada una de las partes del cuerpo y se mantiene actualizado en todo momento con los datos provenientes del SDK.

El siguiente paso en la inicialización es crear una nueva instancia del gesto *Smash* con los valores del archivo de configuración generados previamente en el entrenamiento (línea 3). Una vez cargado el gesto, el desarrollador define la acción asociada al reconocimiento del mismo, es decir, el método que *GRTool* ejecutará cuando el *Smash* sea reconocido (línea 4). Aquí, la suscripción al evento se hace por medio del nombre del método a ejecutar (*GestoReconocidoSmash*) el cual es definido por el desarrollador. Notar que *GRTool* utiliza un sistema de registro de gestos que funciona asincrónicamente a la aplicación. Cuando un gesto es detectado, *GRTool* ejecuta el método correspondiente pasando el control a la aplicación.

El paso final de la inicialización es comenzar el reconocimiento del gesto (línea 5). A partir de ese momento, *GRTool* observa los movimientos del usuario provenientes de Kinect y los almacena temporalmente utilizando el mecanismo de buffering. A medida que el buffer se actualiza, se extrae la trayectoria de la mano derecha y se evalúa la técnica de reconocimiento entrenada. Si el valor resultante de esta evaluación cae dentro del umbral de aceptación, *GRTool* ejecuta el método asociado al reconocimiento del *Smash* (línea 8). Dentro de este método, el desarrollador debe escribir el código específico de las acciones que se toman cuando el gesto es reconocido, siendo este caso la ejecución del movimiento *Smash* y la simulación del golpe a la pelota (línea 9). Adicionalmente, cuando GRTool ejecuta el método envía dos parámetros adicionales *Feedback* y *Animacion*. El parámetro *Feedbak* es una estructura que depende el tipo de técnica utilizada. Si el desarrollador utiliza la técnica HMM el *Feedback* contiene la probabilidad con la que el gesto fue reconocido mientras que para el caso de DTW o Procrustes contiene la distancia, el factor de escala y la rotación aplicada para reconocer el gesto. Por otro lado, el parámetro *Animación* es la colección de frames extraída del buffer que al evaluarla dio como resultado el gesto reconocido.

De forma paralela al reconocimiento del *Smash*, la aplicación muestra la interfaz gráfica del ambiente del juego. Para esto, el desarrollador codifica dentro del método *Update* (línea 6) el comportamiento de los elementos que se muestran. Este método es ejecutado por Unity3D periódicamente durante toda la ejecución de la aplicación. En particular, en la línea 7 se codificó la visualización de los movimientos del usuario por medio de un avatar que utiliza las posiciones del esqueleto actualizadas por *GRTool*.

Algoritmo 1 Reconocimiento del gesto *Smash* utilizando *GRTool*

```
private Esqueleto esqueleto;
private int usuario = 1;
private GameObject avatar;

1:void Start()
 {
  // Definición del usuario a observar.
2: esqueleto = GRTool.AdministradorEsqueletos.obtener-
  Instancia().obtenerEsqueleto(usuario);
  // Inicialización del Smash.
3: Gesto Smash = Gesto.obtenerGesto("Smash");
4: Smash.GestoReconocido += new Gesto.Controlador-
  EventoGestoReconocido(GestoReconocidoSmash);
5: Smash.comenzarReconocimiento();
 }

6:void Update()
 {
  // Actualización en las posiciones del avatar desde Kinect.
7: moverAvatar(esqueleto, avatar);
 }

8:public void GestoReconocidoSmash(Feedback feedback,
Animacion a)
{
  // Se ejecuta la acción asociada al reconocimiento del Smash.
9: ejecutarSmash(avatar);
}
```

Una vez que se ejecuta la aplicación, el jugador se mueve frente a Kinect provocando que las posiciones de las partes del cuerpo del avatar se actualicen (línea 7). Paralelamente, *GRTool* utiliza esas posiciones para evaluar sus movimientos y tratar de identificar el *Smash*. Cuando el gesto es detectado, *GRTool* llama a la aplicación por medio del método registrado *GestoReconocidoSmash* y la aplicación ejecuta la acción esperada por el usuario correspondiente al reconocimiento del *Smash*.

Es importante notar que si bien el ejemplo fue implementado sobre el motor Unity3D, *GRTool* puede ser incluido en cualquier motor de gráfico que soporte lenguaje C# siguiendo los pasos de registro y notificación de eventos.

3 Evaluación

Con el fin de evaluar *GRTool*, se analizaron dos aspectos: la precisión de las técnicas de reconocimiento y el esfuerzo de desarrollo. La precisión de las técnicas de

reconocimiento se midió en términos de la cantidad de gestos reconocidos sobre gestos evaluados. El esfuerzo de desarrollo se midió en base a dos diferentes métricas tomadas del código necesario para implementar el reconocimiento de gestos: la cantidad de líneas de código, y la complejidad ciclomática. La cantidad de líneas de código considera sólo sentencias sin blancos ni comentarios. La complejidad ciclomática calcula el número de caminos de ejecución diferentes dentro de un fragmento de código. A continuación se presentan los experimentos para evaluar la precisión del reconocimiento y el esfuerzo de desarrollo.

3.1 Precisión de las técnicas de reconocimiento

Para evaluar la precisión de las técnicas se creó una base de gestos de entrenamiento compuesta por los gestos *Smash*, *Círculo*, *Nadar* y *Estiramiento de pierna*; descriptos en la Fig. 5. La elección de los mismos se debe a que involucran partes superiores e inferiores del cuerpo. Cada gesto fue realizado 50 veces por dos personas de diferente contextura física y en distintas posiciones dentro del campo de detección de Kinect.

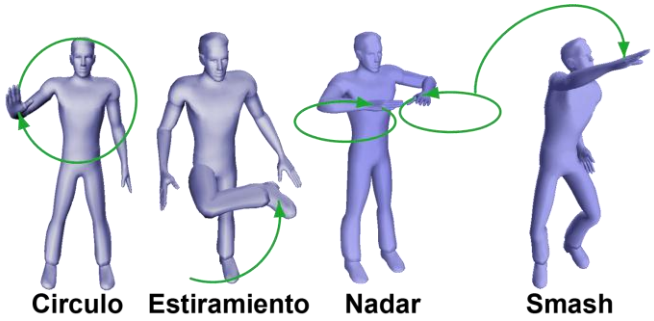


Fig. 5. Trayectorias de los gestos utilizados en los experimentos

Tomando la base de gestos de entrenamiento, se variaron las cantidades de ejemplares utilizados para entrenar cada técnica de reconocimiento con los cuatro gestos, entre 5, 25, 50 y 75 muestras por cada gesto. Al finalizar cada entrenamiento se evaluaron los 25 ejemplares restantes de cada gesto como conjunto de prueba y se contabilizaron los gestos reconocidos correctamente. Para el caso de la técnica HMM que requiere definir a priori la cantidad de clústeres a utilizar, se la entreno con 5, 10, 15 y 20 clústeres a fin de determinar el impacto de la cantidad de clústeres en el reconocimiento. La Fig. 6 muestra los resultados del experimento agrupados en cantidad de gestos reconocidos siendo 100 la cantidad óptima de reconocimiento (25 *Círculo* + 25 *Estiramiento* + 25 *Nadar* + 25 *Smash*).

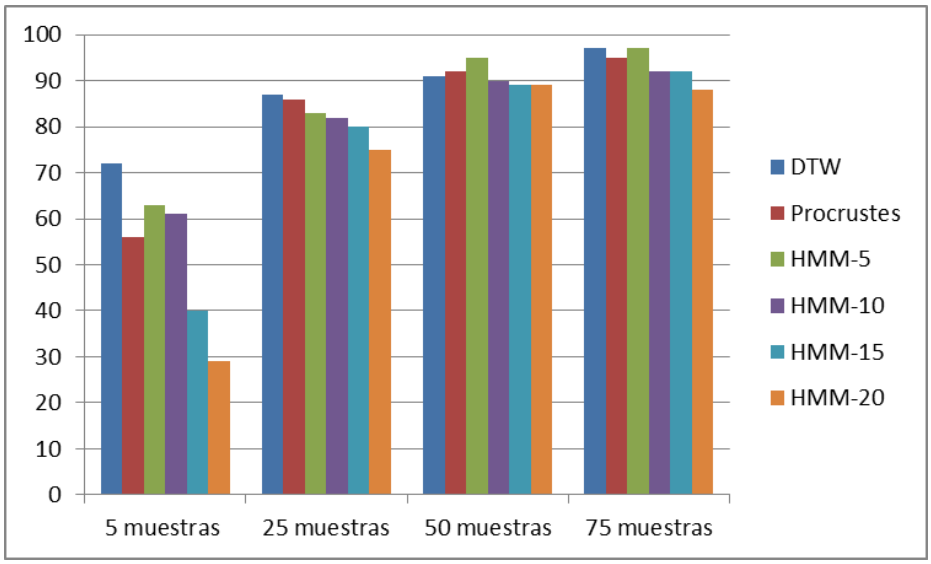


Fig. 6. Gráfico comparativo de la cantidad de gestos reconocidos por cada técnica variando la cantidad de muestras utilizadas para el entrenamiento

La figura muestra que aumentar la cantidad de ejemplares utilizados en el entrenamiento influye favorablemente en la precisión de reconocimiento de todas las técnicas. Por ejemplo, DTW reconoce 72 gestos cuando se la entrena con 5 ejemplares y 97 gestos cuando se utilizan 75 muestras, aumentando un 34.7% la precisión del reconocimiento. Por otro lado, se puede apreciar que aumentar la cantidad de clústeres para la técnica HMM hace que disminuya la cantidad de gestos reconocidos. Esto se debe a que al aumentar la cantidad de clústeres se aumenta la fidelidad del gesto a reconocer, haciendo que la imitación del gesto sea más difícil de realizar. En otras palabras, el gesto a reconocer debe pasar por cada uno de los clústeres para ser correctamente reconocido.

3.2 Esfuerzo de desarrollo

Para evaluar el esfuerzo de desarrollo de una aplicación con y sin *GRTool*, se utilizó la aplicación *Uso de Gestos para Exploración de Modelos 3D* (UGEM3D) implementada directamente sobre el SDK y se re-implementó usando *GRTool*. Esta aplicación permite visualizar modelos 3D y manipularlos mediante los gestos que se muestran en la Fig. 7.

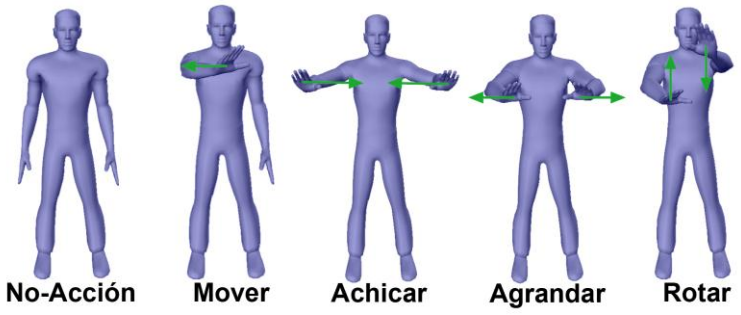


Fig. 7. Gestos reconocidos por la aplicación *Uso de Gestos para Exploración de Modelos 3D*

La implementación original del reconocimiento de gestos para UGEM3D se realizó por medio de condiciones en sentencias *IF* sobre las posiciones de las partes del cuerpo en el espacio 3D. Al cumplirse las condiciones se ejecutaban las acciones definidas.

Tomando de base la implementación de UGEM3D, se cambió la especificación original del reconocimiento de los gestos por otra que utiliza *GRTool* y se compararon las soluciones. Esta comparación se realizó por medio de las métricas cantidad de líneas de código y complejidad ciclomática. El resultado de este experimento se muestra en la Fig. 8.

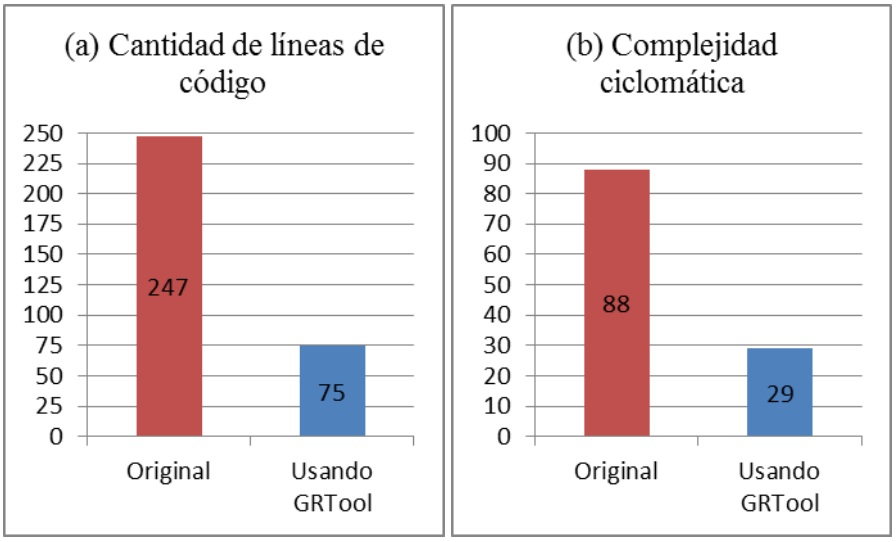


Fig. 8. Resultados de la comparación de cantidad de líneas de código (a), y comparación de complejidad ciclomática (b) con y sin *GRTool*

El gráfico (a) muestra que usar *GRTool* permite un ahorro del 69,63% en comparación con la versión original, en términos de la cantidad de líneas de código. Además, el gráfico (b) muestra un disminución de un 67,04% en la complejidad ciclomática al

utilizar *GRTool*. Estas mejoras se deben a que el reconocimiento se encuentra encapsulado dentro de la herramienta y el desarrollador sólo debe escribir el código que se ejecutará cuando cada gesto sea reconocido.

3.3 Lecciones aprendidas y limitaciones

Las evaluaciones anteriores muestran que utilizar *GRTool* permite de forma sencilla incluir reconocimiento de gestos en aplicaciones de terceros con precisiones mayores al 95%. Se ha comprobado que incrementar la cantidad de ejemplares utilizados en el entrenamiento mejora la precisión de las técnicas de reconocimiento. Además, el incremento de la cantidad de clústeres utilizados en la técnica HMM mejora la exactitud de la detección del gesto del usuario. Sin embargo, cuando más exacta es la técnica, más exacto debe ser el gesto realizado por el usuario para que sea reconocido. Por lo tanto, el desarrollador debe encontrar un balance entre la exactitud de la técnica y el grado de usabilidad de la aplicación.

Con respecto al esfuerzo de desarrollo, la re-implementación de *Uso de Gestos para Exploración de Modelos 3D* por medio de *GRTool* consiguió una reducción de un 69,63% de líneas de código en relación a la versión inicial de la aplicación. Aún más, el código necesario para reconocer gestos utilizando *GRTool* posee un 67,04% menos de complejidad ciclomática que sin la herramienta.

En resumen, los resultados obtenidos en ambas evaluaciones de *GRTool* son prometedores y sería de gran importancia realizar nuevas pruebas con un conjunto más variado de gestos, aplicaciones y otras herramientas que facilitan el uso de Kinect con el fin de aumentar la generalización de los resultados.

4 Conclusión

En este trabajo se presentó *GRTool*, una herramienta que facilita la tarea de los desarrolladores al momento de crear aplicaciones de interfaz natural usando el dispositivo Kinect. La misma ofrece soporte para la creación, entrenamiento y reconocimiento de gestos. Para las dos primeras actividades, *GRTool* provee una interfaz gráfica que abstrae al desarrollador de cualquier tipo de conocimiento técnico específico de los algoritmos de reconocimiento de gestos. En cuanto a la etapa de reconocimiento, el desarrollador dispone de un código template para incluir el reconocimiento de los gestos entrenados dentro de su aplicación, reduciendo de esta manera el esfuerzo de desarrollo.

Los experimentos realizados sobre la herramienta generaron resultados prometedores. La evaluación de la precisión de las técnicas en el reconocimiento de gestos obtuvo resultados superiores al 95%. Además, *GRTool* permitió obtener una reducción del 69,63% en líneas de código y un 67,04% en complejidad ciclomática con respecto a una implementación similar de una aplicación basada en gestos contando únicamente con el SDK de Kinect.

Un aspecto que se podría integrar a *GRTool* es el soporte para un reconocimiento de gestos adaptable inteligentemente a las capacidades motrices de la persona que

realiza el gesto. Por ejemplo, el desarrollador podría desear que el reconocimiento sea más restrictivo en cuanto a la exactitud del movimiento cuando el usuario posee una gran destreza o menos restrictivo cuando el usuario carece de dicha habilidad. Si bien *GRTool* no posee un ajuste automático, el desarrollador puede crear dos gestos por separado, adecuando cada uno a las destrezas de cada tipo de usuario.

Referencias

1. T. Schlömer, B. Poppinga, N. Henze and S. Boll. Gesture Recognition with a Wii Controller. Proceedings of the Second International Conference on Tangible and Embedded Interaction (TEI'08), Bonn, Germany, 2008.
2. D. Zufferey.Device based gesture recognition.Written as part of a master seminar about gesture recognition. Department of Informatics (DIUF), University of Fribourg, Fribourg, Switzerland, 2009.
3. J.C Lee. Hacking the Nintendo Wii Remote. Pervasive Computing, IEEE, vol.7, no.3, pp.39-45, 2008.
4. S. Kean, J. Hall and P. Perry. Meet the Kinect: An Introduction to Programming Natural User Interfaces. Apress, 2011.
5. F. Kistler, B. Endrass, I. Damian, C. Dang and E. André. Natural interaction with culturally adaptive virtual characters. Journal on Multimodal User Interfaces. Springer-Verlag, vol.6, no.1-2, pp.39-47, 2012.
6. E. A. Suma, B. Lange, A. Rizzo, D. M. Krum and M. Bolas. Faast: The flexible action and articulated skeleton toolkit. In Virtual Reality Conference. IEEE. pp.247-248, 2011.
7. V. Thiruvarudchelvan and T. Bossomaier. Towards realtime stance classification by spiking neural network. Neural Networks (IJCNN). The 2012 International Joint Conference on. IEEE, pp.1-8, 2012.
8. A. Bleiweiss, D. Eshar, G. Kutliroff, A. Lerner, Y. Oshrat and Y. Yanai. Enhanced interactive gaming by blending full-body tracking and gesture animation. In ACM SIGGRAPH ASIA 2010 Sketches. pp.34. 2010.
9. S. Salvador and P. Chan. FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space. Intelligent Data Analysis, IOS Press, vol.7, no.11, pp.561-580, 2007.
10. L. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE, vol.77, no.2, pp.257-286, 1989.
11. A. Ross. Procrustes Analysis. Course report, Department of Computer Science and Engineering, University of South Carolina, 2004.
12. T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman and A. Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. Pattern Analysis and Machine Intelligence. IEEE Transactions, vol.24, no.7, pp.881-892, 2002.

Apéndice A

La Fig 9 muestra un archivo de configuración que corresponde al antrenamiento de dos gestos (*Circulo* y *Smash*) utilizando la técnica DTW. Cada gesto está definido por un nombre, la ruta donde se encuentran los gestos utilizados en el entrenamiento y los valores de distancia entre gestos obtenidos del entrenamiento. Para que un nuevo gesto sea reconocido, la distancia obtenida de la evaluación debe ser menor al valor de *Distancia Maximo*. Ademas, el desarrollador dispone de otros indicadores como son los valores de *Distancia Minimo*, *Mediana* y *Promedio*. Estas distancias son calculadas luego de obtener la distancia de cada gesto en el conjunto de entrenamiento. La *Mediana* se calcula ordenando las distancias de menor a mayor y tomando como indicador el valor central. Mientras que, para calcular el *Promedio* se suman todas las distancias y se divide el resultado por la cantidad de gestos dentro del conjunto de entrenamiento. Todos estos indicadores pueden ser modificados manualmente por el desarrollador. Particualarmente, modificar el valor de *Distancia Maximio* cambia el umbral de aceptación de la técnica haciendola más o menos restrictiva.

```
<Tecnica Tipo="DTW">
 <Gestos>
  <Gesto Nombre="Circulo"
    Ruta="Entrenamiento\Circulo\">
   <ParteCuerpo Nombre="ManoIzquerda" />
   < Distancia Minimo="2.235289812088" Maximo="4.69306993484"
   Mediana="3.27742433547974" Promedio="3.46417987346649" />
  </Gesto>
  <Gesto Nombre="Smash"
    Ruta="Entrenamiento\Smash\">
   <ParteCuerpo Nombre="ManoDerecha" />
   <Distancia Minimo="3.7172372341156" Maximo="10.8299751281"
   Mediana="6.4306960105896" Promedio="7.27360618114471" />
  </Gesto>
 </Gestos>
</Tecnica>
```

Fig. 9. Archivo de configuración XML para un entrenamiento de la técnica DTW

En la ruta indicada en el archivo de configuración XML, se encuentran todos los ejemplares del gesto utilizados para el entrenamiento. La Tabla 1 muestra el formato un gesto *Smash* almacenado. Cada $celda_{ij}$ de la tabla es la posicion en el espacio de la parte del cuerpo i en el frame j siguiendo el formato (X, Y, Z). Observando cada fila individualmente se obtiene la trayectoria que describe cada parte del cuerpo a lo largo del tiempo. Particularmente, la trayectoria de la mano derecha es la única requerida para reconocer el gesto *Smash* según indica el archivo de configuración.

	Frame 1	Frame 2 …
Cabeza	(-0.3306832, 0.4002154, 2.625942)	(-0.3337381, 0.3972753, 2.625359)
Cuello	(-0.2891778, 0.19429, 2.627669)	(-0.2882654, 0.1920274, 2.623773)
Torso	(-0.2517311, -0.0155933, 2.635672)	(-0.2469307, -0.0171828, 2.628801)
Hombro Izq.	(-0.3881992, 0.171582, 2.495469)	(-0.3873129, 0.16928, 2.491533)
Codo Izq.	(-0.4439334, 0.03821963, 2.222167)	(-0.4414717, 0.03560876, 2.216588)
Mano Izq.	(-0.2944176, 0.2004035, 2.055722)	(-0.2965088, 0.2009123, 2.049705)
Hombro Der.	(-0.1901563, 0.2169979, 2.75987)	(-0.189218, 0.2147749, 2.756013)
Codo Der.	(-0.1419175, 0.0250853, 2.960983)	(-0.1375318, 0.03973025, 2.9662)
Mano Der.	(0.07007173, 0.1849407, 3.054248)	(0.04961444, 0.2253999, 3.062448)
Cintura Izq.	(-0.2746978, -0.2393309, 2.563019)	(-0.2659972, -0.240265, 2.553186)
Rodilla Izq.	(-0.3288592, -0.6513672, 2.432086)	(-0.3159774, -0.6525233, 2.421113)
Pie Izq.	(-0.3661192, -1.064129, 2.465354)	(-0.3597662, -1.064257, 2.459239)
Cintura Der.	(-0.153871, -0.2116225, 2.724331)	(-0.1451948, -0.2125212, 2.714471)
Rodilla Der.	(-0.2272892, -0.6410492, 2.713665)	(-0.2241696, -0.6410217, 2.71129)
Pie Der.	(-0.204937, -1.00635, 2.806846)	(-0.2023975, -1.011061, 2.810197)

Table 1. Gesto smahs almacenado en la base de gestos

Apéndice B

El modo de funcionamiento de la aplicación UGEM3D puede interpretarse como una máquina de estados en la cual el estímulo para el cambio de estado es el reconocimiento de un gesto (Fig. 9).

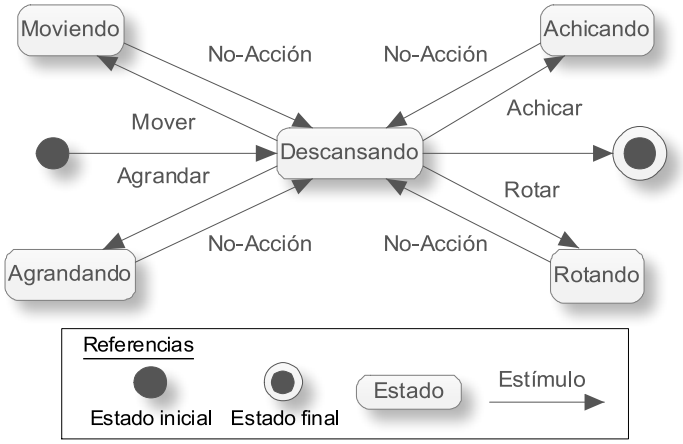


Fig. 10. Máquina de estados de la ejecución de UGEM3D

Inicialmente la aplicación comienza en el estado Descansando y cuando alguno de los 4 gestos es detectado, la aplicación cambia al estado correspondiente al gesto reconocido. Aquí, la aplicación ejecuta la función que modifica las propiedades del modelo 3D hasta detectar el gesto No-Acción, volviendo nuevamente al estado Descansando. Las figuras 11, 12, 13 y 14 muestran la interfaz gráfica de *Uso de Gestos para Exploración de Modelos 3D*.

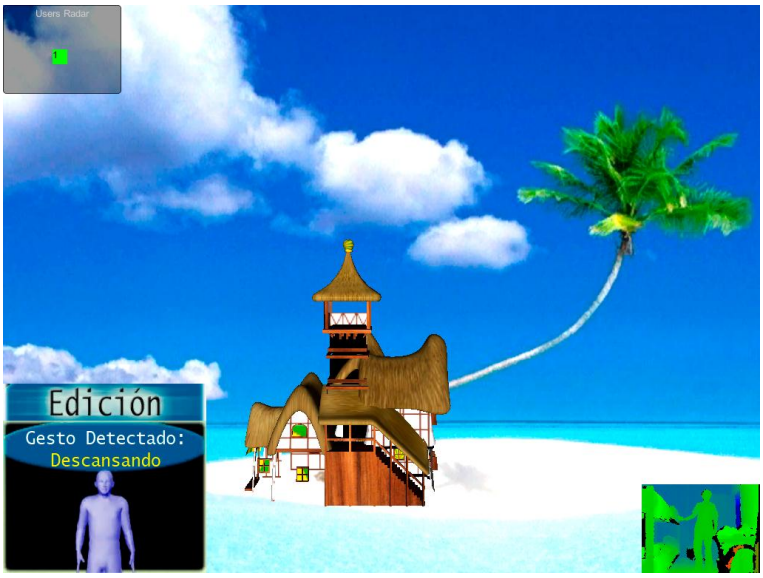


Fig. 11. Interfaz gráfica de UGEM3D al detectar el gesto descansando

Fig. 12. Interfaz gráfica de UGEM3D al detectar el gesto agrandar

Fig. 13. Interfaz gráfica de UGEM3D al detectar el gesto mover

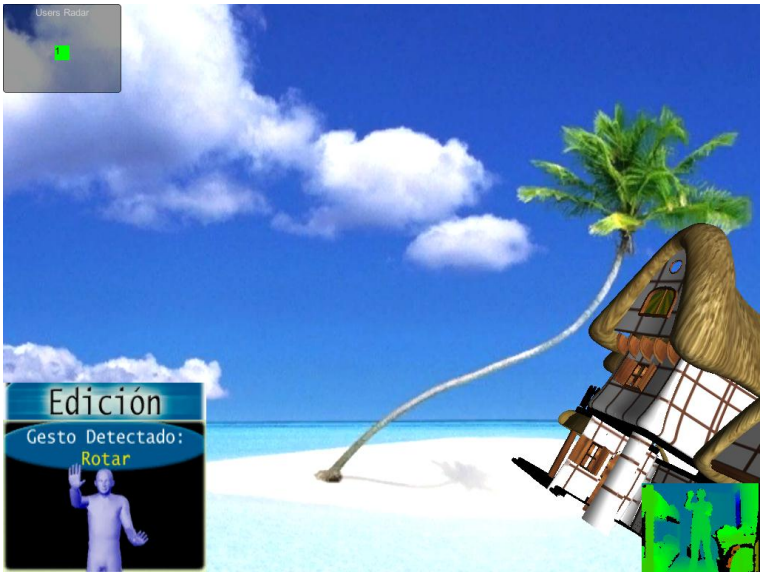


Fig. 14. Interfaz gráfica de UGEM3D al detectar el gesto rotar